determining which of the identified user customizations are compatible with the second modified version; and

creating the upgrade version of the software application by merging the compatible user customizations with the second modified version.

29. (New) The method of claim 28 wherein the identifying of the user customizations in the first modified version that are not present in the second modified version includes comparing objects resident in the first and second modified versions.

30. (New) The method of claim 28, further comprising:

determining identified user customizations that conflict with the second modified version;

automatically notifying a user of the conflicting user customizations; and

presenting options to the user for resolving the conflicting user customizations.

31. (New) A computer-implemented method of creating a new version of a software application by merging a first modified version of a software application and a second modified version of the software application, the method comprising:

automatically determining a set of differences between the first modified version and the second modified version;

automatically determining which differences in the determined set are compatible differences; and

creating a new version of the software application by modifying the second modified version to include the compatible differences.

32. (New) The method of claim 31 including automatically determining differences in the determined set that are conflicting differences.

33. (New) The method of claim 32 including automatically resolving one or more of the conflicting differences without user intervention based at least in part on

predefined conflict resolution rules, and wherein the automatic determining of the compatible differences includes selecting the resolved conflicting differences.

34. (New) The method of claim 32 including notifying a user of at least some of the conflicting differences.

35. (New) The method of claim 34 including presenting options to the user for resolving at least some of the conflicting differences of which the user was notified and resolving one or more of the conflicting differences in a manner based on one or more of the presented options selected by the user.

36. (New) The method of claim 34 including determining an importance of each of one or more of the conflicting differences, and wherein the notifying of the user of conflicting differences includes providing notification of the determined importances.

37. (New) The method of claim 32 wherein the first and second modified versions include one or more common objects, and wherein the conflicting differences include a difference in attributes of one of the common objects between the first and second modified versions.

38. (New) The method of claim 32 wherein the first and second modified versions each include objects, and wherein the conflicting differences include a difference in the objects present in the first and second modified versions.

39. (New) The method of claim 31 wherein the differences determined to be compatible include conflicting differences that are identified as superficial.

40. (New) The method of claim 31 wherein the differences determined to be compatible include conflicting differences corresponding to objects that are not identified as being of high priority.

41. (New) The method of claim 31 including, before the automatic determining of the set of differences, displaying to a user a graphical user interface related to creating a new version of the software application, and wherein the automatic determining of the set of differences is in response to an indication from the user via the graphical user interface.

42. (New) The method of claim 41 wherein the displaying of the graphical user interface to the user includes displaying an option to include differences between the first and second modified versions in the determined set only if the differences correspond to high priority aspects of the first and second modified versions and/or displaying an option to abort creating a new version if a specified number of errors is exceeded.

43. (New) The method of claim 41 wherein the first and second modified versions of the software application have a common ancestor version of he software application, wherein one of the modified versions has been customized by a user so as to delete one or more aspects of the common ancestor version, wherein the aspects of the common ancestor version deleted in the one modified version have not been deleted in the other modified version, and including displaying an option to the user to select the deleted aspects as being a compatible difference such that the created new version does not include the deleted aspects.

44. (New) The method of claim 31 including displaying to a user at least some of the differences of the determined set so as to allow visual comparison of those two versions.

45. (New) The method of claim 31 wherein at least one of the modified versions includes modifications made by a user and/or at least one of the modified versions includes modifications made by a developer of the software application.

46. (New) The method of claim 31 wherein the automatic determining of the set of differences is further between the first and second modified versions and a common ancestor version.

47. (New) The method of claim 31 wherein the first modified version includes user modifications to a prior version of the software application, wherein the second modified version is an upgrade for that prior version, and wherein the creating of the new version is to upgrade the software application to a version of the upgrade that includes at least some of the user modifications.

48. (New) The method of claim 47 wherein the automatic determining of the set of differences and the automatic determining of the compatible differences comprises:

determining a first set of differences based on a comparison of the user-modified prior version and the prior version;

determining a second set of differences based on a comparison of the upgrade version and the prior version; and

determining which differences from the first set are compatible with the second set of differences.

49. (New) The method of claim 48, wherein the user-modified prior version comprises a first plurality of objects and the prior version comprises a second plurality of objects and the upgrade version comprises a third plurality of objects, and wherein the determining of the first and second sets of differences comprises:

determining one or more objects from the first plurality of objects that share a common name with one or more objects from the second plurality of objects;

for each of at least one of the objects from the first plurality of objects that shares a common name with one of the objects from the second plurality of objects, determining one or more attributes associated with that object that differs from the attributes associated with the object that shares the common name and including data related to the difference between those attributes in the first set of differences;

determining one or more objects from the second plurality of objects that share a common name with one or more objects from the third plurality of objects; and

for each of at least one of the objects from the second plurality of objects that shares a common name with one of the objects from the third plurality of objects, determining one or more attributes associated with that object that differs from the attributes associated with the object that shares the common name and including data related to the difference between those attributes in the second set of differences.

50. (New) The method of claim 47 wherein the user-modified prior version comprises a first plurality of objects, the prior version comprises a second plurality of objects, and the upgrade version comprises a third plurality of objects, and wherein the determining of the set of differences comprises determining that an object from the first plurality of objects is not included within the second or third plurality of objects and indicating that that object is a compatible difference.

51. (New) The method of claim 47 wherein the user-modified prior version comprises a first plurality of objects, the prior version comprises a second plurality of objects, and the upgrade version comprises a third plurality of objects, and wherein the determining of the set of differences comprises determining that an object from the second and third plurality of objects is not included within the first plurality of objects and indicating that absence of that object from the first plurality of objects is a conflicting difference.

52. (New) A computer-readable medium whose contents cause a computing device to perform a method for upgrading at least portions of a modified version of a software application based at least in part on another distinct modified version of the software application, the method comprising:

automatically determining a set of differences between a first modified version of a software application and a second modified version of the software application, the differences in the determined set including at least portions of the first modified version that differ from at least portions of the second modified version;

automatically determining differences in the determined set that are compatible; and

upgrading the first second version by including at least some of the compatible differences.

53. (New) The computer-readable medium of claim 52 wherein the first and second modified versions each include associated objects, and wherein the portions of the first modified version that differ from the portions of the second modified version are objects associated with the first modified version that differ from objects associated with the second modified version.

54. (New) The computer-readable medium of claim 52 wherein the first and second modified versions share a common ancestor version, and wherein the automatic determining of the set of differences includes comparing at least two of the first modified version, the second modified version and the common ancestor version.

55. (New) The computer-readable medium of claim 52 wherein the contents that cause the computing device to perform the method is executable software code.

56. (New) An apparatus for upgrading a software application from a user-modified prior version to an upgrade version, the user-modified prior version and the upgrade version having a common ancestor version, the apparatus comprising:

a first component able to determine a set of differences between the user-modified prior version and at least one of the upgrade version and the common ancestor version, the determining based on a comparison of the user-modified prior version to the upgrade version and/or the common ancestor version;

a second component able to identify differences in the determined set that are compatible with the upgrade version; and

a third component able to apply changes to the upgrade version based on the compatible differences.

57. (New) A computing device for upgrading a software application from a user-modified prior version to an upgrade version, wherein the user modified prior version and the upgrade version have a common ancestor version, said apparatus comprising:

means for determining a first set of differences based on a comparison of the user modified prior version and the common ancestor version and for determining a second set of differences based on a comparison of the upgrade version and the common ancestor version;

means for determining which differences from said first and second sets of differences are compatible differences and which are conflicting differences; and

means for applying changes to the upgrade version associated with said compatible differences.

58. (New) A method of upgrading a software application from a user-modified prior version to an upgrade version, wherein the user-modified prior version and the upgrade version have a common ancestor version, said method comprising:

comparing the user-modified prior version, the common ancestor version, and the upgrade version to determine differences;

determining which of the differences are compatible; and

applying changes to the upgrade version associated with the compatible differences.

59. (New) The method of claim 58 wherein the user-modified prior version comprises a first plurality of objects, the common ancestor version comprises a second plurality of objects, the upgrade version comprises a third plurality of objects, and wherein the comparing comprises comparing the first plurality of objects with the second and third pluralities of objects to determine whether objects were added, deleted, or modified by a user.

60. (New) The method of claim 59 wherein the determining of the compatible differences includes indicating that differences associated with objects added by the

user are compatible differences and that differences associated with objects modified by the user are compatible differences if the modified objects are superficial.

61. (New) The method of claim 59 including determining conflicting differences by indicating that differences associated with objects deleted by the user are conflicting differences.

62. (New) A computer-implemented method of migrating user changes between versions of a software application, the method comprising:

identifying user changes to a first version of a software application; and

automatically migrating at least some of the identified user changes to a distinct second version of the software application by,

determining at least some of the identified user changes as being compatible with the second version of the software application; and

using the determined compatible user changes when executing the second version of the software application.

63. (New) The method of claim 62 wherein the determining of the user changes that are compatible with the second version includes analyzing types of the user changes.

64. (New) The method of claim 62 wherein the user changes are customizations of the first version.

65. (New) The method of claim 64 wherein the customizations include changes to one or more of objects used by the first version.

66. (New) The method of claim 64 wherein the customizations include changes to resource files used by the first version.

67. (New) The method of claim 64 wherein the customizations include changes to one or more of help files used by the first version, database tables used by the first version, database table contents used by the first version, reports used by the first version, and string tables used by the first version.

68. (New) The method of claim 62 wherein the user changes are new modules added to the first version.

69. (New) The method of claim 62 wherein the user changes are new objects added to the first version and/or deletions of existing objects from the first version.

70. (New) The method of claim 62 wherein the user changes are configurations of the first version.

71. (New) A computer-implemented method of assisting a user in customizing a new version of a software application based on prior user changes, the method comprising:

identifying prior user changes to a software application;

displaying to the user indications of at least some of the identified user changes; and

after receiving one or more selections from the user, automatically customizing the new version to include identified user changes indicated by the user selections.

72. (New) The method of claim 71 including automatically identifying some of the identified prior user changes as compatible with the new version and customizing the new version to include those compatible user changes.

73. (New) The method of claim 71 including automatically identifying some of the identified prior user changes as conflicting with the new version, and wherein

indications of the conflicting user changes that are displayed to the user include indications that those user changes are conflicting.

74. (New) A method of assisting a user in comparing differences between two software applications to assist in usage of at least one of the software applications, the method comprising:

comparing the two software applications to determine differences;

automatically determining compatibility of at least some of the differences; and

displaying to a user indications of at least some of the determined differences in such a manner as to include indications of determined compatibility, so as to assist the user in determining whether to use one of the software applications.

75. (New) The method of claim 74 wherein one of the two software applications is a software application currently in use and the other of the two software applications is a distinct other software application, and wherein the comparing of the differences between the two software applications is to test an upgrade from the current software application to the distinct other software application.

76. (New) A method of assisting in migrating between a current version and a new version of a software application, the method comprising:

defining conflict resolution rules to assist in resolving differences between versions of a software application, the defining based at least in part on input from a user; and

automatically assisting in migrating between the current version and the new version of the software application by,

comparing the current and new versions to determine differences;

using one or more of the defined conflict resolution rules to resolve at least some of the determined differences; and

applying at least the resolved differences to the current version.